Messaging

in distributed systems

micro-23

Aliaksei Bialiauski

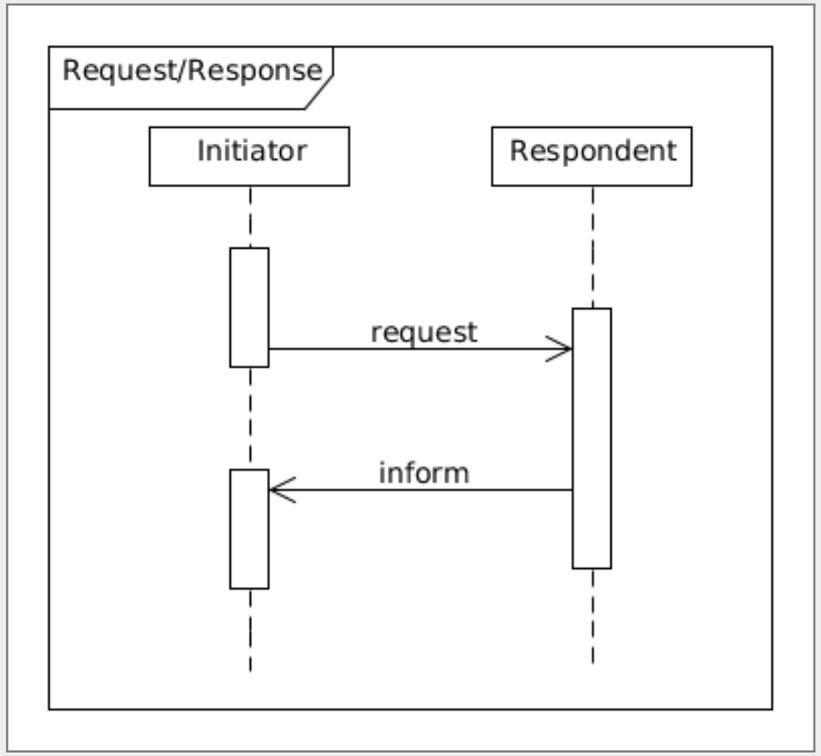
Designed in LaTeX

- Request-Response vs Producer-Consumer
- Messaging Patterns
- Internals of messaging

Chapter #1:

Request-Response vs Producer-Consumer

Request-Response



Availability

What should the client do when the server is not available?

Performance

What should the client do when the server gets slower?

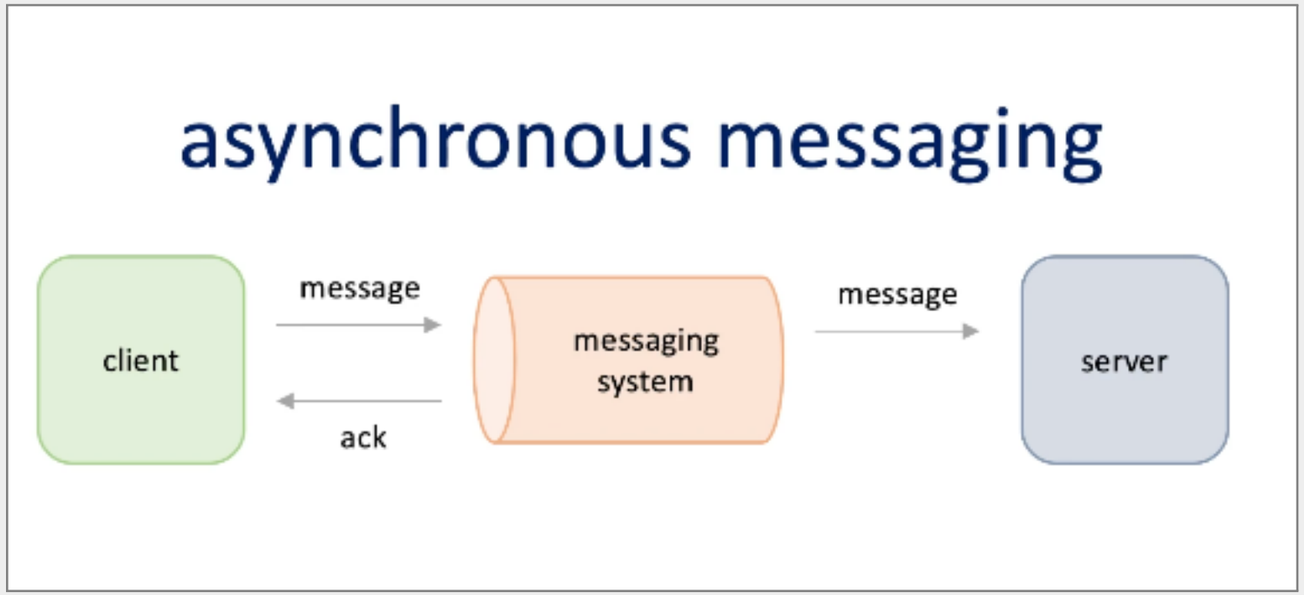
Backpressure

What should the server do when there is a sudden traffic spike?

Retry

What should the client do with failed requests?

Asynchronous messaging



`Availability` -> Keep messages and send them later

`Performance` -> Add more servers to parallelize processing

`Backpressure` -> Keep draining messages at its own pace

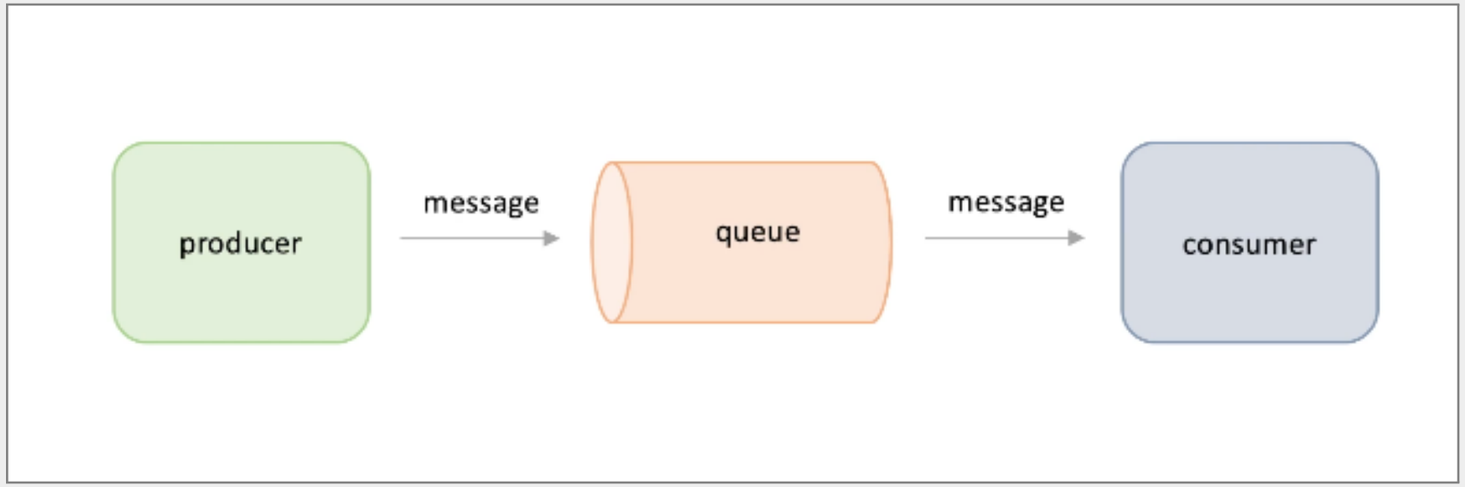
`Retry` -> Re-send failed messages

Also, asynchronous messaging helps to `decouple` client and server.

Chapter #2:

Messaging Patterns

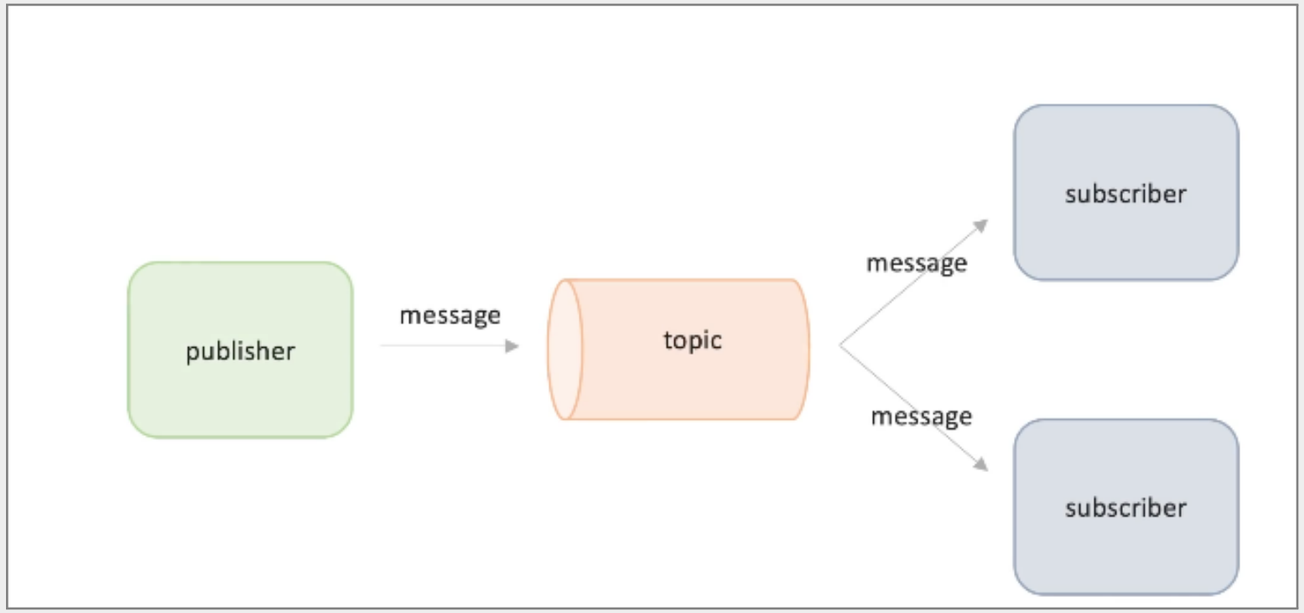
Message Queuing



https://aws.amazon.com/sqs

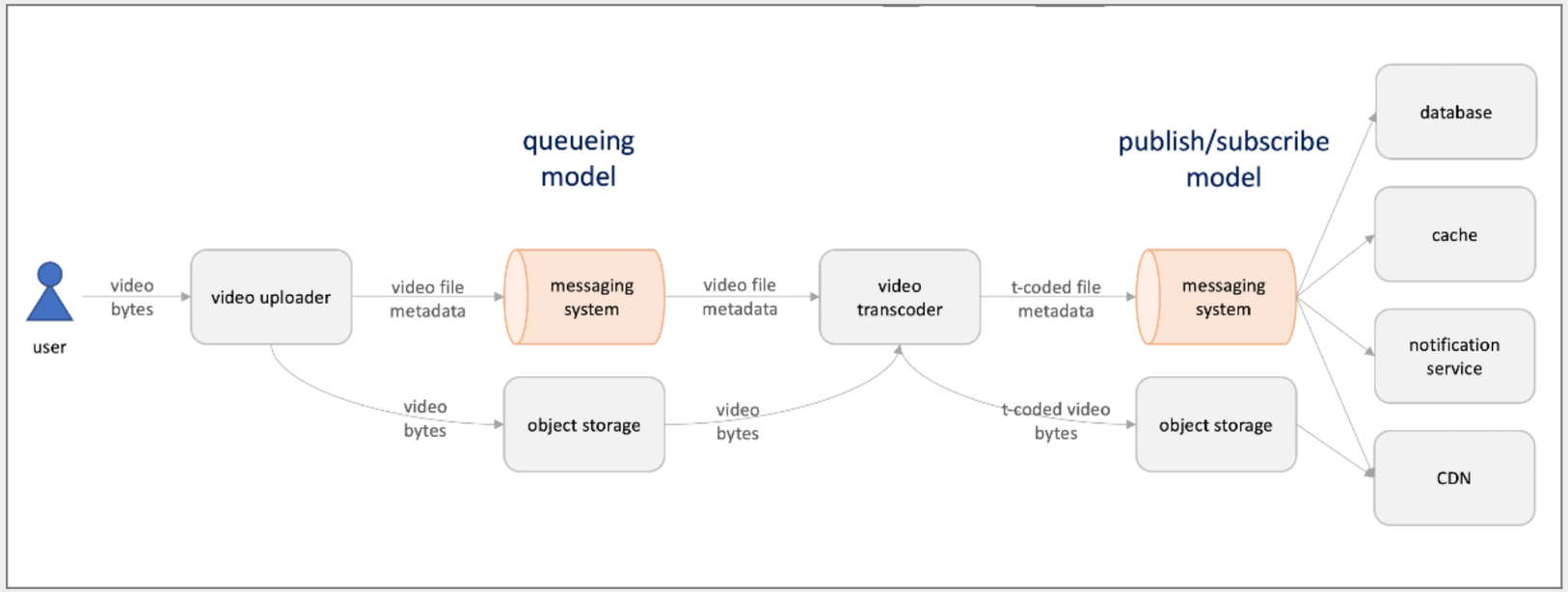
https://www.rabbitmq.com

Publish-Subscribe

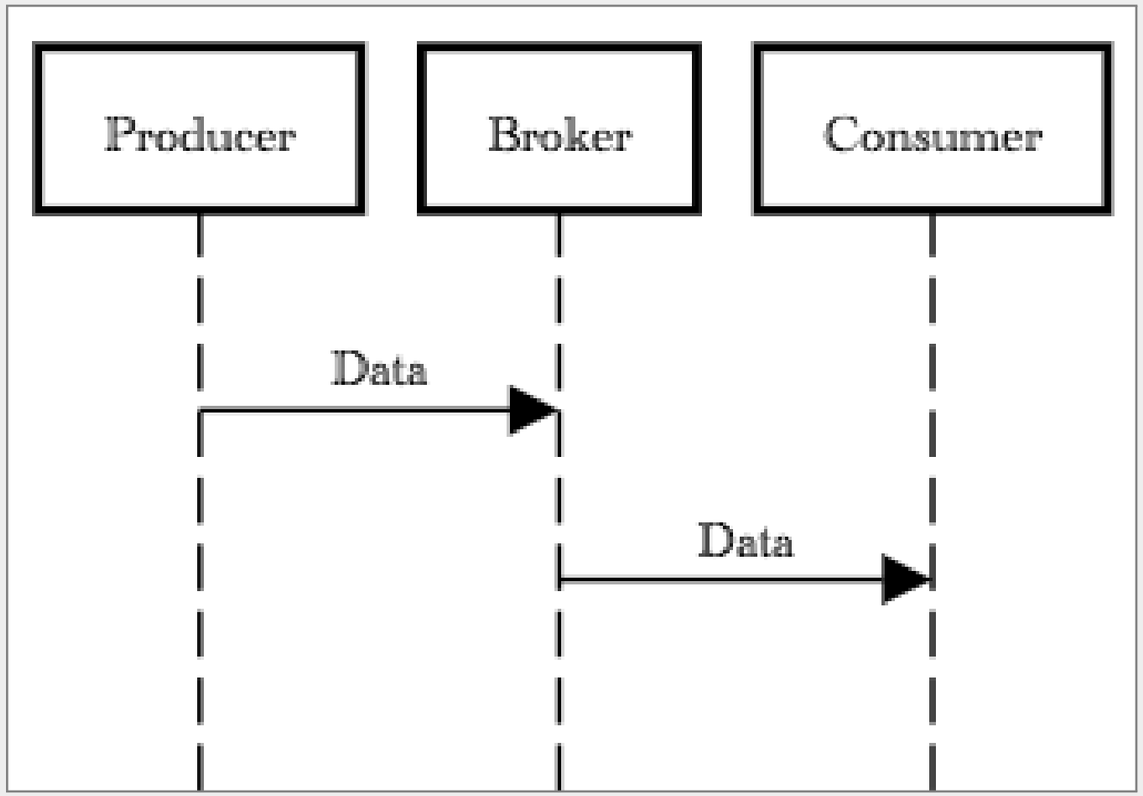


https://kafka.apache.org

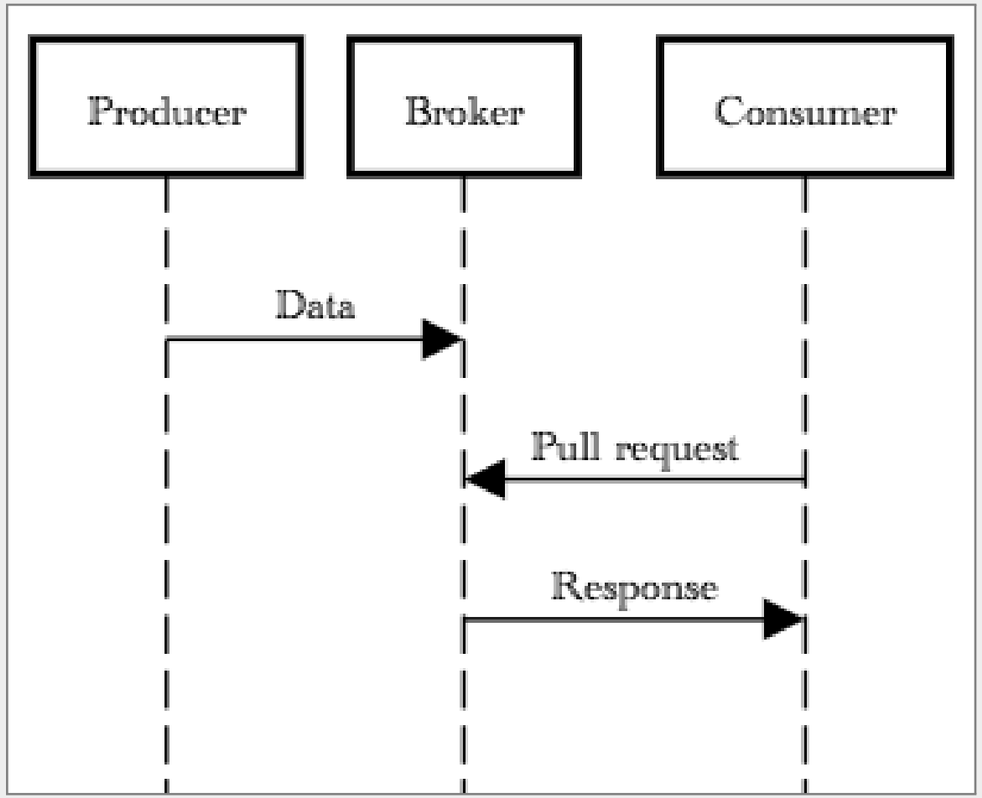
https://aws.amazon.com/kinesis


queueing model
publish/subscribe model
user
video bytes
video uploader
video file metadata
messaging system
video file metadata
video transcoder
t-coded file metadata
messaging system
database
cache
notification service
CDN
video bytes
object storage
video bytes
t-coded video bytes
object storage

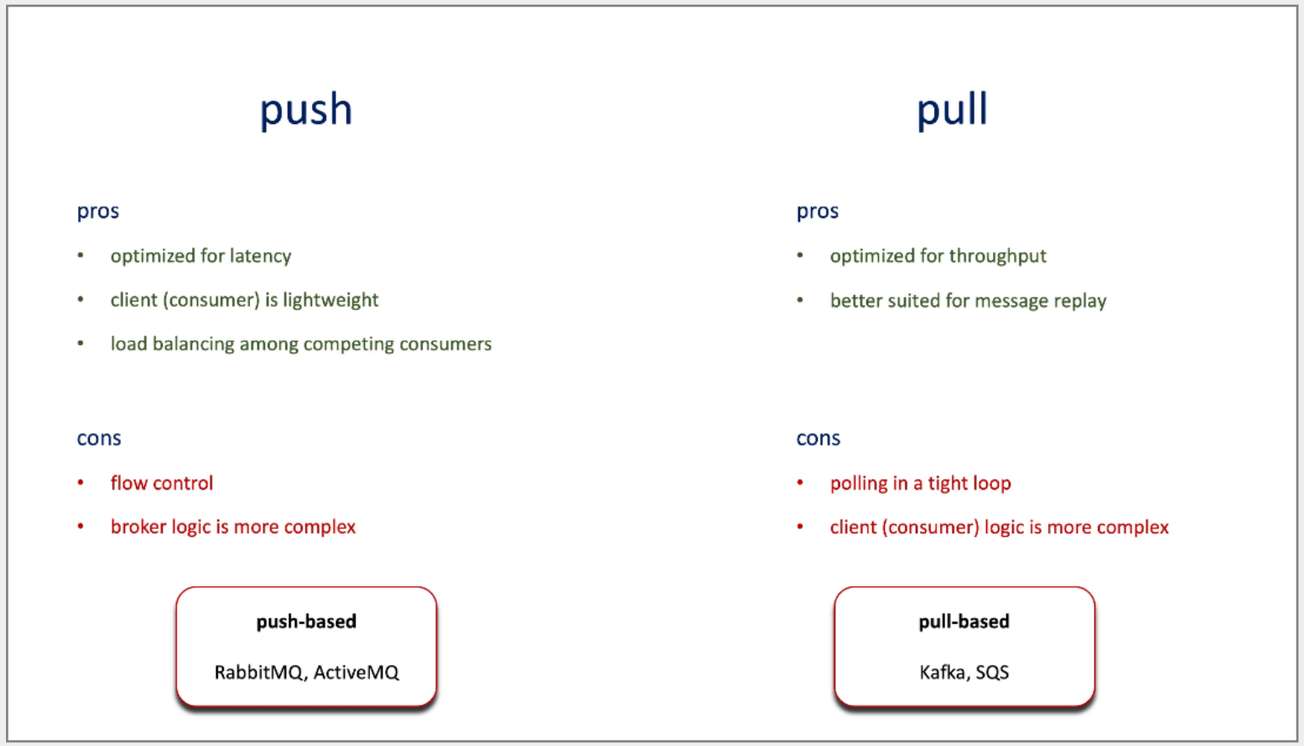
Push model



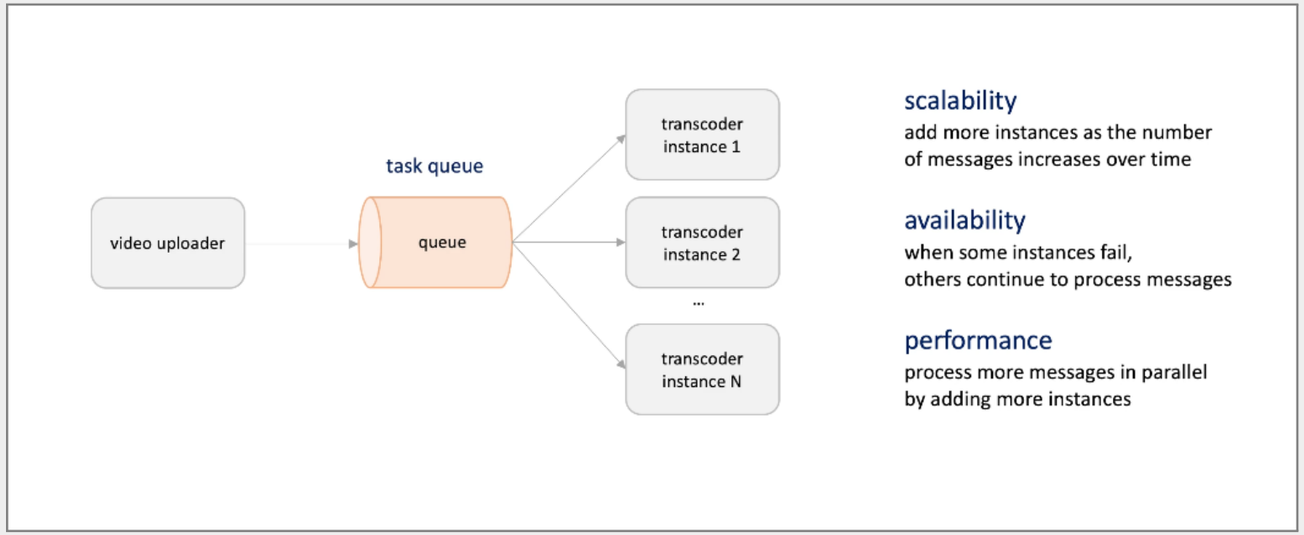
Pull model



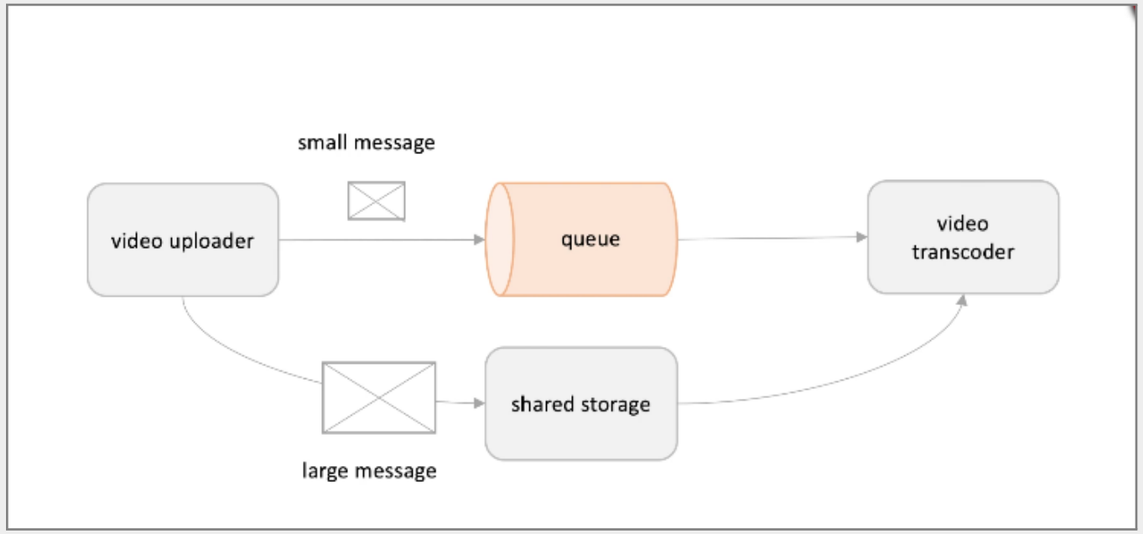
Push vs Pull


push
pros
optimized for latency
client (consumer) is lightweight
load balancing among competing consumers
cons
flow control
broker logic is more complex
push-based
RabbitMQ, ActiveMQ
pull
pros
optimized for throughput
better suited for message replay
cons
polling in a tight loop
client (consumer) logic is more complex
pull-based
Kafka, SQS

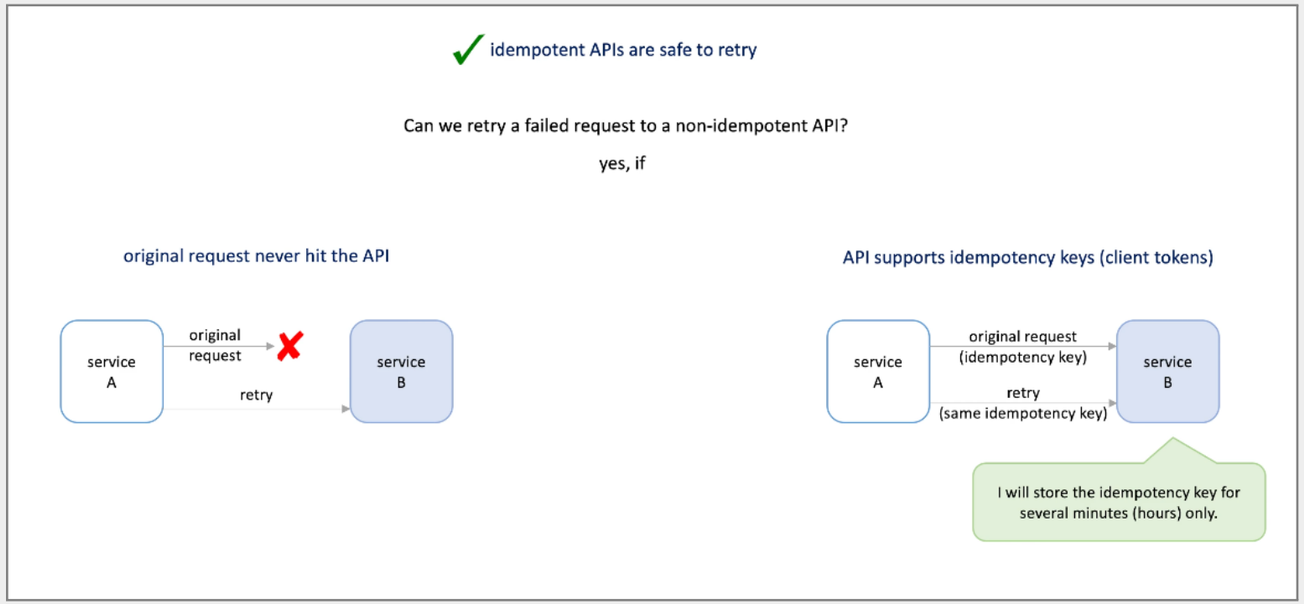
Competing Consumers



Claim Check


small message
video uploader
queue
video transcoder
shared storage
large message

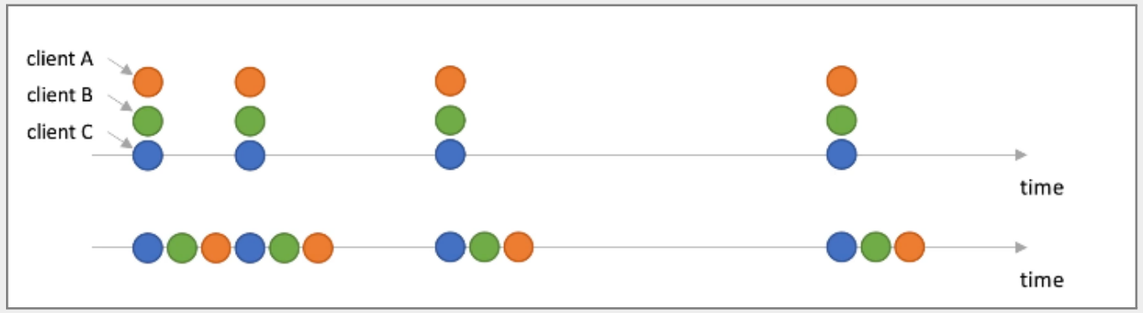
When to retry?



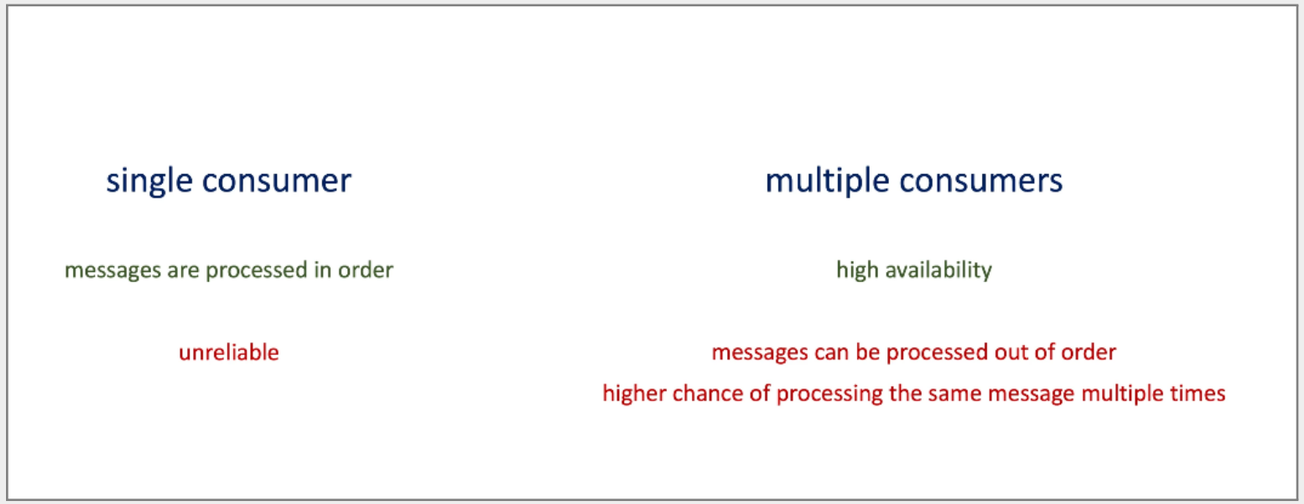
HTTP Status Code	Description	Retry?	Comments
N/A	Network connection is not established.	Yes	It doesn't matter if the API is idempotent or not. If no connection was established, the request was never sent.
400	Bad request.	No	There is a problem with the request. For example, a required parameter is not supplied for the specified action. Retry will not help.
403	Access denied.	No	One of many possible reasons - the provided security credentials are not valid.
404	Not found exception.	No	There is no requested resource present on the server side. E.g. the specified S3 bucket does not exist.
429	Too many requests.	Yes (but after a delay)	The request was not processed. It was immediately rejected by the server due to the high request rate.
500	Internal server error.	Yes (if idempotent)	The service encountered an unexpected condition that prevented it from fulfilling the request.
503	Service unavailable.	Yes	The requested service is not available.

Exponential backoff

Retry at different times

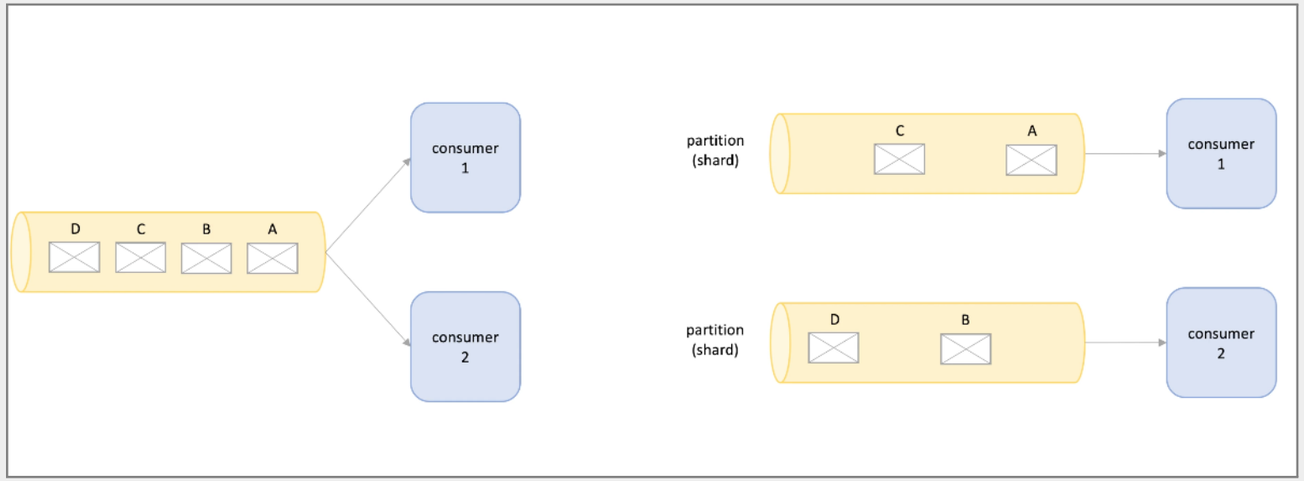


Sigle consumer vs Multiple consumers

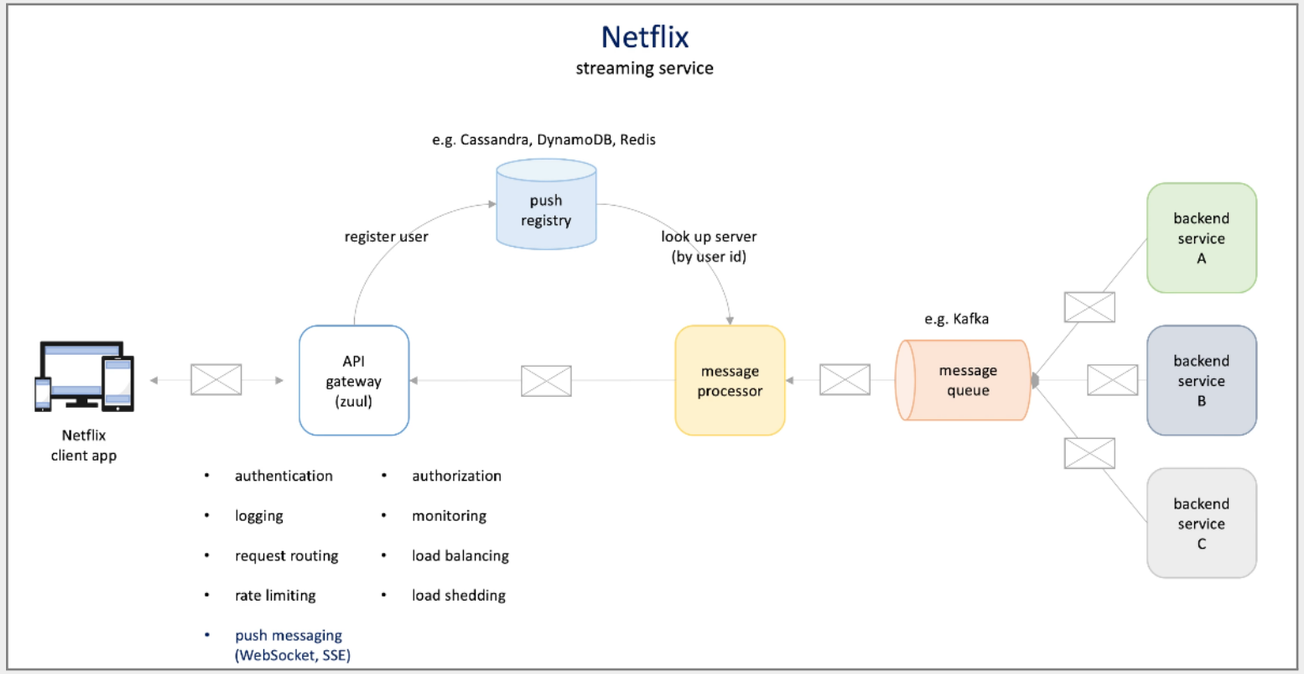


Multiple consumers break the order of messages

Data partitioning



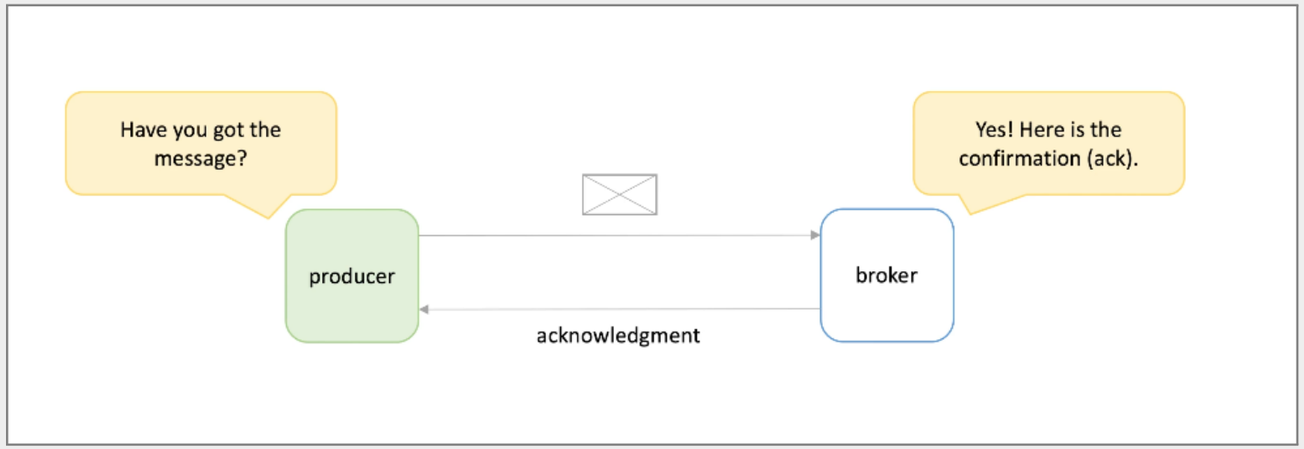
Netflix architecture



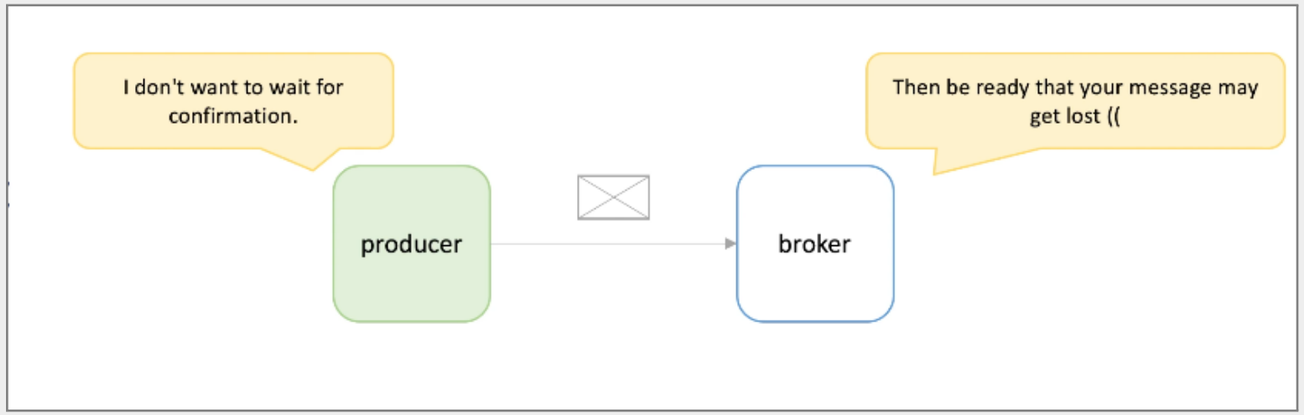
Chapter #3:

Internals of messaging

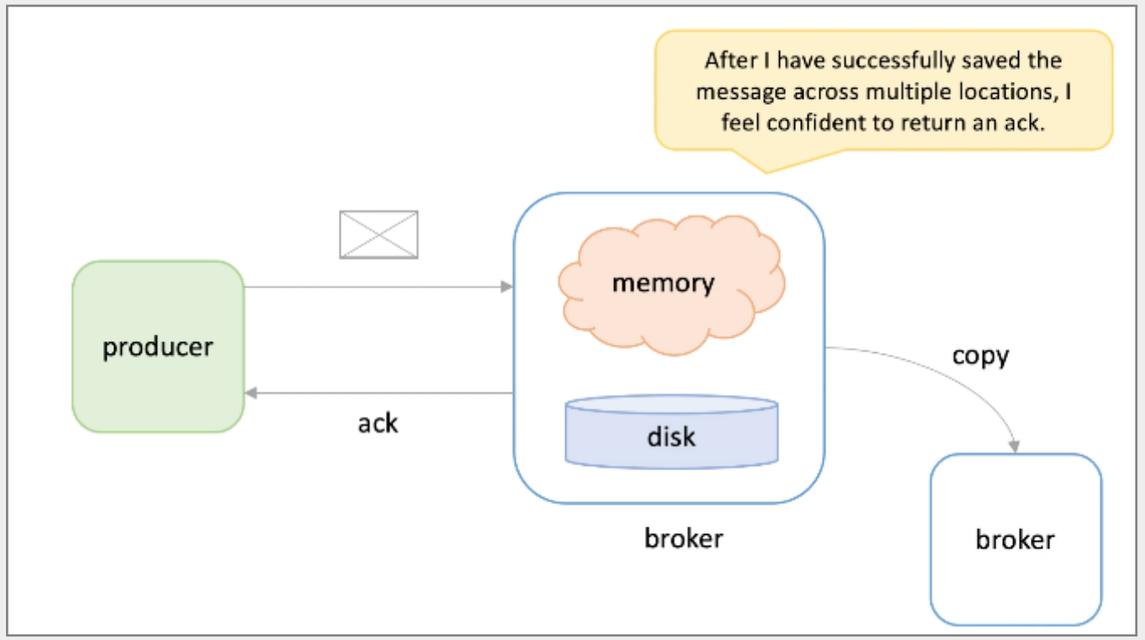
Message Acknowledgement



Unsafe ack aka “fire and forget”

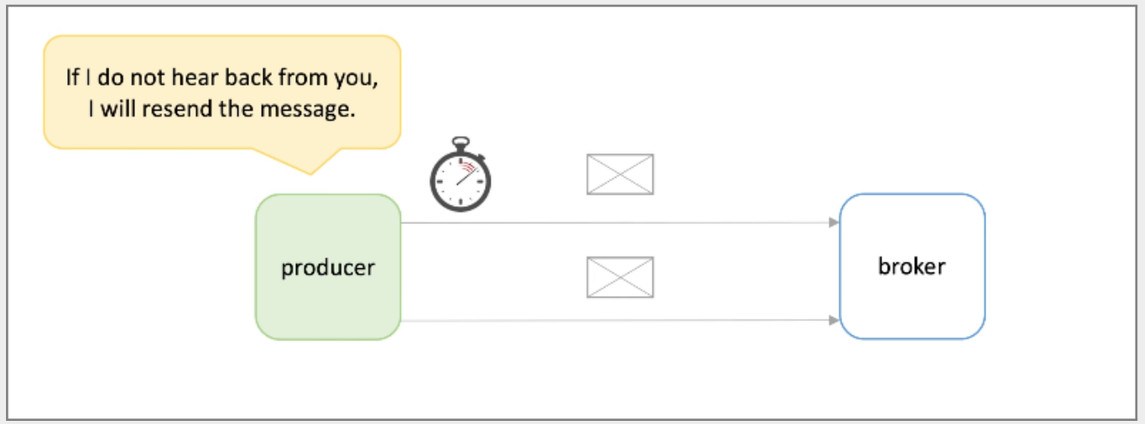


Safe ack

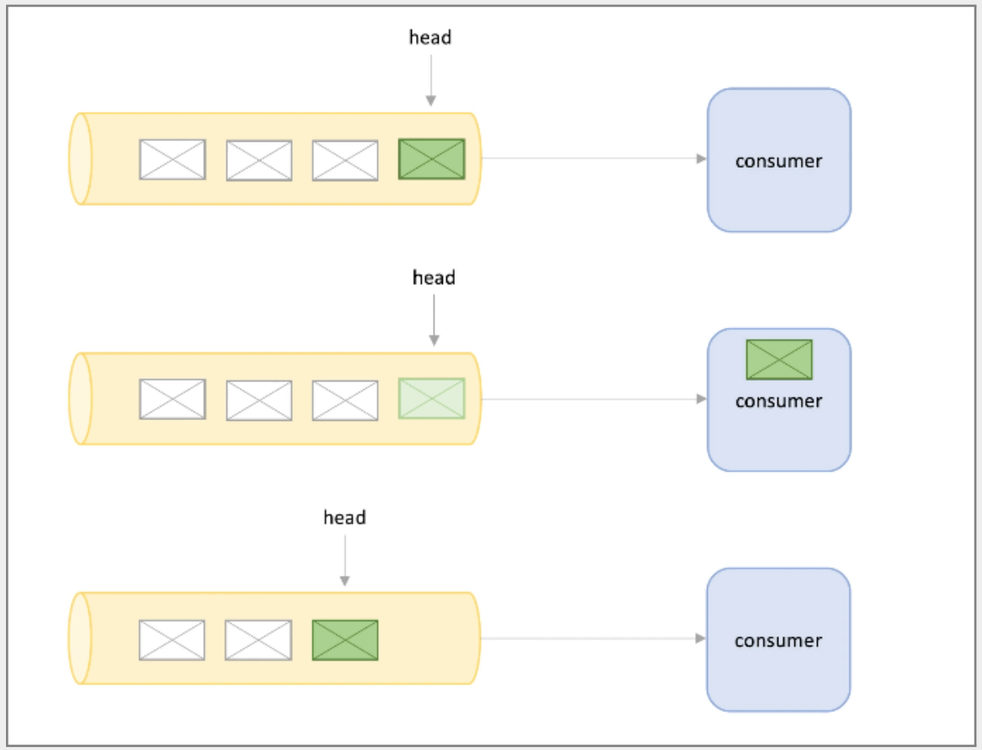


It is up to the broker what safe really means

Failed ack

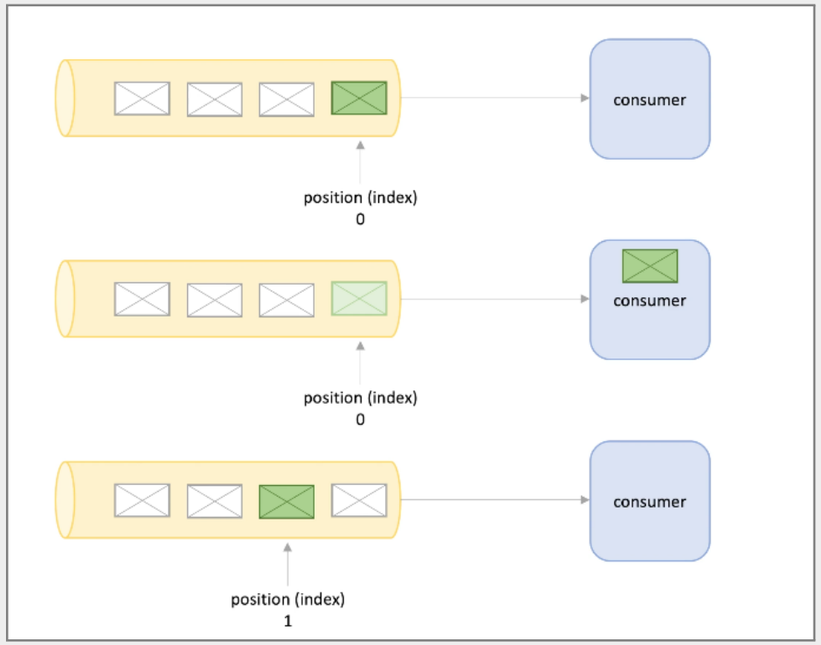

If I do not hear back from you,
I will resend the message.
producer
broker

Consumer Offsets - Head delete



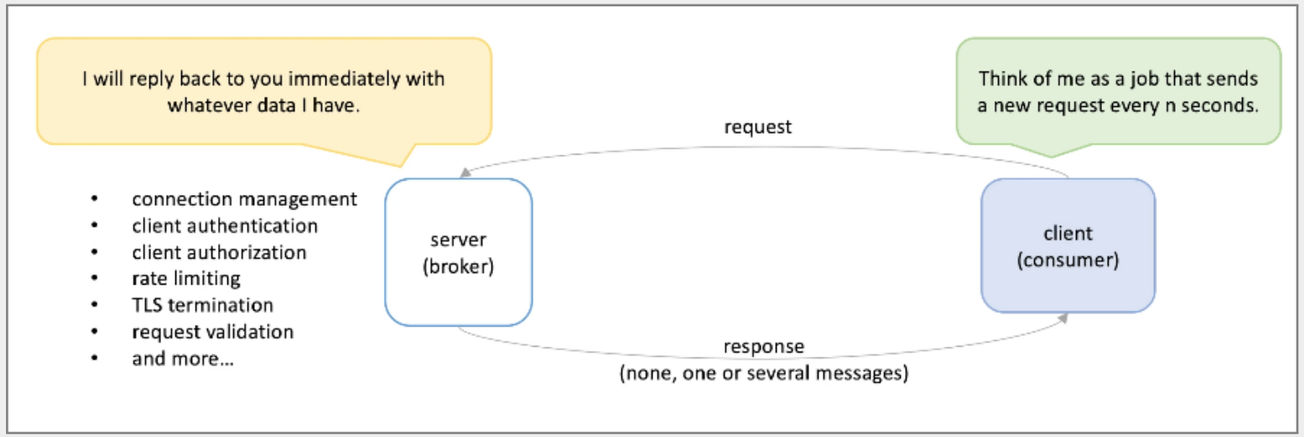
RabbitMQ, SQS

Incrementing the pointer

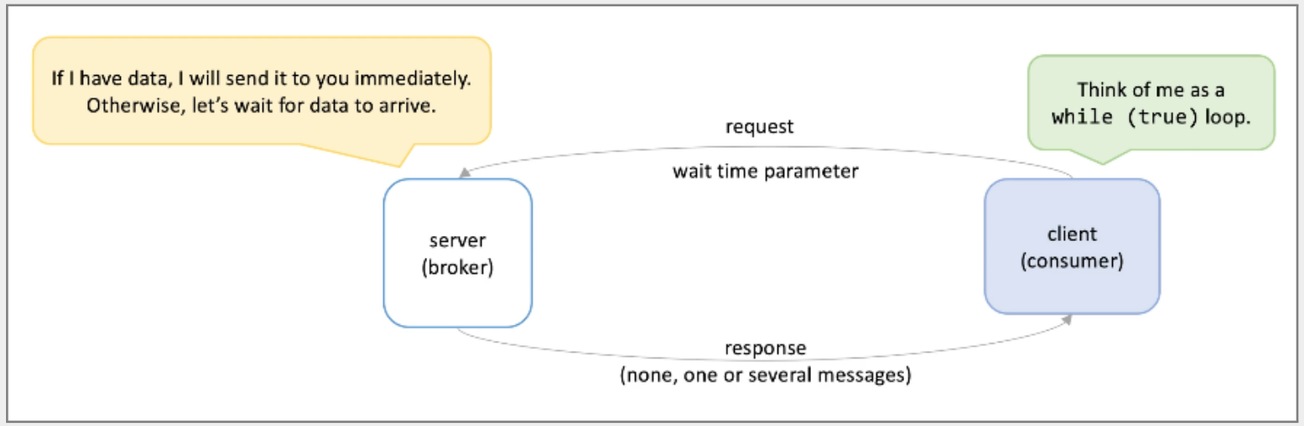


Kafka, Kinesis

Short Polling



Long Polling



Delivery Guarantees

`at-most-once` -> Messages might be lost

`exactly-once` -> Messages processed only one time

`at-least-once` -> Messages migth be processed multiple times

But failures will happen!

“ Failures are a given and everything will eventually fail over time: from routers to hard disks, from operating systems to memory units corrupting TCP packets, from transient errors to permanent failures. This is a given, whether you are using the highest quality hardware or lower cost components. ”

— Dr. Werner Vogels

Most of the messaging systems you will work with will be configured to provide `at-least-once` delivery guarantee